PALADYN Journal of Behavioral Robotics

VERSITA

Research Article · DOI: 10.2478/s13230-010-0001-5 · JBR · 1(1) · 2010 · 1-13

A Formalism for Learning from Demonstration[*]

Erik A. Billing[†], Thomas Hellström[‡]

Department of Computing Science,
Umeå University, Umeå, Sweden

Received 12 October 2009

Accepted 26 February 2010

Abstract

The paper describes and formalizes the concepts and assumptions involved in *Learning from Demonstration (LFD)*, a common learning technique used in robotics. LFD-related concepts like *goal, generalization*, and *repetition* are here defined, analyzed, and put into context. Robot behaviors are described in terms of trajectories through information spaces and learning is formulated as mappings between some of these spaces. Finally, behavior primitives are introduced as one example of good *bias* in learning, dividing the learning process into the three stages of *behavior segmentation*, *behavior recognition,* and *behavior coordination*. The formalism is exemplified through a sequence learning task where a robot equipped with a gripper arm is to move objects to specific areas. The introduced concepts are illustrated with special focus on how bias of various kinds can be used to enable learning from a single demonstration, and how ambiguities in demonstrations can be identified and handled.

Keywords

learning from demonstration · ambiguities · behavior · bias · generalization · robot learning

1. Introduction

Learning From Demonstration (LFD) is a well established technique for teaching robots how to perform useful tasks. The basic idea is that the robot learns a behavior from one or several demonstrations performed by a, most often human, teacher. The research area is attractive, both in its intuitive approach to human robot interaction and as a framework for a theoretical analysis of knowledge representation and transfer of knowledge between intelligent agents.

Research on LFD is influenced by a variety of fields, including control theory, artificial intelligence, psychology, ethology, and neuro physiology. While primarily being a big asset, the multidisciplinary nature of LFD also contributes to the lack of a unified formalism for the different components constituting the research field. It should not come as a surprise that the terminology used differs for works conducted by researchers from various areas. In this paper, we define and formalize the common ideas and principles involved in LFD. The presented work is both a survey of how these concepts are used in research, and an attempt to describe them in the light of related concepts in machine learning, planning theory, and psychology. To our knowledge this has not been previously done in a unified way and the result can be used both as a theoretical introduction to the field and as framework for further development and research. In contrast to other surveys of the area [4, 12], the present work specifically focuses on LFD where the robot is directly controlled during demonstration, e.g. via teleoperation or kinematic teaching. While this direction removes some of the hard and important issues in LFD, it allows increased focus on other aspects, specifically how bias is introduced into the LFD process.

The formalism is applied to a sequence learning task in which the introduced concepts are illustrated with a special focus on how bias of various kinds can be used to enable learning from a single demonstration, and how ambiguities in demonstrations can be handled.

The formal approach is inspired by the work on planning and actuation by LaValle [53] and therefore does not always follow the terminology and notation found in common literature on LFD. Where this is the case, it is highlighted and the commonly used terms are referred.

In Section 2, a few fundamental concepts that form the basis for the rest of the paper are introduced. Section 3 gives a formal description of the learning process using these concepts. In Section 4, the introduced formalism is applied on a sequence learning task using a Khepera robot equipped with a gripper arm. Section 5 summarizes the paper and discuss directions for future research. A symbol index summarizing introduced notations can be found in Table 5.

2. Basic concepts

2.1. State space

One fundamental component in classical AI is the concept of a *state space* X, described by a *world ontology* [77, p.222]. The state space can be defined as a set of all possible situations that could arise in the world [53, p.17]. More specifically, the state space only includes the *relevant* aspects of the world, given a particular task or limited set of tasks. However, if the task is unknown it is very difficult to identify which aspects of the world are relevant. One could of course try to include all aspects that might be of interest, but even if possible, that would result in a huge and complex state space, implying tremendous sensing requirements when applied to a field such as LFD. Furthermore, defining a state space introduces many unnecessary assumptions about the world, and requirements for information which make the problem much more complex than necessary. This observation is nicely illustrated by

[*] Parts of this text also appear as a technical report: E. A. Billing and T. Hellström. Formalising Learning from Demonstration, *UMINF 08.10*, Department of Computing Science, Umeå University, Sweden, 2008.
[†] E-mail: billing@cs.umu.se
[‡] E-mail: thomash@cs.umu.se

Simons' ant [81] and is also related to the *frame problem* [47, 62].

For these reasons, it is desirable to create new spaces, less task-specific and sensor-demanding, in which behaviors can be represented. Such a redefined representation is referred to as an *information space* [53, ch.11]. The concept of information spaces is also common within LFD, but appears under different names. In order to facilitate learning, approaches to LFD often utilize so called *primitives* or *skills*. These primitives can be seen as building blocks from which more complex behaviors can be composed, which results in moving the learning process away from the state space into a new representational space composed of the available skills, e.g. [8, 34, 46, 51, 65, 68]. Many of these approaches relate strongly to *Behavior Based Control (BBC)* [5, 58, 60]. BBC has its roots in the reactive paradigm, but emphasizes parallel, loosely connected *behaviors* for control of the robot as an emergent property, rather than a single stimuli-response loop.

The possibility of applying the concept of information spaces within LFD is further investigated in Section 3, but first a few other basic concepts have to be introduced.

2.2. Sensing and acting

Imagine an agent interacting with the environment. It perceives the world through its sensors and acts upon the world with its actuators. The sensors are defined as a function $h : X \rightarrow Y$ transforming a state $x \in X$ into a sensor state $y \in Y$ [53, p. 561]. Y denotes the *observation space*, i.e., the set of all possible readings returned by the agent's sensors. Each $y \in Y$ is a vector $(y\,(1)\,, y\,(2)\,, ...)$ comprising simultaneous values from all sensors. Typical examples are a thermometer that maps physical temperatures x to numbers $y\,(1) \in \mathbb{R}$ or a GPS receiver that maps physical positions to latitude and longitude, $y\,(2) \in \mathbb{R}^2$. Y corresponds to the *stimulus domain* in behavior-based robotics [5].

On the actuator side, actions can be said to transform a state into another state. Hence, actuators implement the function $f : X \times U \rightarrow X$ where U denotes the *action space*, i.e., the set of all possible actions the agent can execute. A typical example is the requested velocity for each motor of the robot. Note that this does not specify the actual motor velocity, and only the outgoing information is represented in U. The actual velocity is usually represented in state space X.

Now a description of how the agent behaves, i.e. generates actions, can be introduced. In general, such a description is referred to as a *controller*, but is also known as a *plan* [53, p.560], *behavior mapping* [5, 27, 68, 71], *motor primitive* [3], *control policy* [4] or *inverse model* [39]. Several important differences between these terms do exist, for example in terms of abstraction level and temporal extension, but for now they can all be said to implement the function π:

$$\pi : X \rightarrow U. \tag{1}$$

Hence, π maps states $x \in X$ to actions $u \in U$. As mentioned before, X is not explicitly represented in the agent. Still, the physical sensors and actuators can be said to implement the functions h and f, respectively. In contrast, π can not be implemented without an explicit definition of and access to X. To solve this issue, π is later redefined and then controls the agent based on the information space instead of the state space.

2.3. Information space

The observation and action spaces are widely used by the robotics community. These spaces are often combined into a *information space* $I = U \times Y$, also known as the *sensory-motor space* [73]. In each stage k the robot experiences a *sensory-motor event* $e_k = (u_{k-1}, y_k) \in I$. The action at $k-1$ is used since u_k changes the current stage to $k+1$.

One approach that extensively uses representations in I is *sensory-motor coordination (SMC)* [72]. From an SMC perspective, sensing and acting are not two separate processes. In contrast to classical reactive systems, SMC does not view the information flow purely as going from sensors to actuators. Actions give rise to stimuli, just as much as stimuli influences actions. If the agent can predict these relations, it can intentionally control its interactions with the world. Hence, control is seen as a problem of coordination. Similar views are common within psychology, anthropology and cognitive science, [37, 45, 82].

The sensory-motor space I has several advantages when compared to the state space. Most importantly, it is easily defined. If an agent is designed with a fixed number of sensors and actuators, the size of I remains constant independently of environment and task. Of course this limits the possibility of adding new sensors or actuators to the agent without changing the robot's representational space and as a consequence affects previous representations, but for many applications this is a reasonable limitation. The sensory motor space also has a number of drawbacks. In contrast to state space, I does not necessarily contain all information necessary to make a control decision at each moment. A decision, i.e., selection of the next action, may have to be based not on the most recent sensor and motor readings, but on complex patterns of previously observed sensory-motor events. Let $\tilde{Y}_k$ denote the *history observation space*, i.e., the set of all possible observation histories $\tilde{y}_k$ until current stage k:

$$\tilde{y}_k = (y_1, y_2, \ldots, y_k) \in \tilde{Y}_k \tag{2}$$

where each vector $y_i \in Y$ is provided by the sensors at stage i. Similarly, let $\tilde{U}_k$ be the *history action space*, i.e., the set of all possible action histories until current stage k:

$$\tilde{u}_k = (u_1, u_2, \ldots, u_k) \in \tilde{U}_k \tag{3}$$

where each $u_i \in U$ is a particular action vector issued at stage i.

The histories $\tilde{y}_k$ and $\tilde{u}_k$ in combination with the initial conditions η_0 form a *history information state* η_k, also referred to as an *event history*. η_k includes all accumulated information up to stage k [53, p.566]:

$$\eta_k = (\eta_0, \tilde{u}_{k-1}, \tilde{y}_k) \in I_k \tag{4}$$

The initial conditions η_0 describe presumptions about the state of the world X before stage 1. The history information state is a central concept in the formalism since it represents all the information the agent has received, and as a consequence η_k is always known in stage k. I_k is known as the *history information space* and should be understood as the set of all possible event histories up until stage k [53, p.565]:

$$I_k = I_0 \times \tilde{U}_{k-1} \times \tilde{Y}_k \tag{5}$$

where I_0 represents the set of all possible initial conditions.

The definition of I_k becomes impractical in cases where the number of stages is not fixed. Instead, we normally refer to the *information history space* I_{hist}, which has an unspecified length [53, p.657]:

$$I_{hist} = I_0 \cup I_1 \cup I_2 \cup \ldots \tag{6}$$

I_{hist} includes all possible combinations of everything the agent could possibly observe and do. Most $\eta \in I_{hist}$ will of course never appear, due to limitations imposed by the environment and the physical shape of the robot. For example, imagine a simple robot, equipped with a proximity sensor on each of its four sides, placed in an empty large square box. In this environment, the robot never observes a y_k with high activation of all proximity sensors simultaneously. This is a simple consequence of physical properties of the environment and the robot itself. The same reasoning could easily be applied to a human agent. There is a huge amount of patterns the human senses theoretically could perceive, but only a fraction of these will actually be observed.

Most of the formal definitions in this paper take place in history information space I_{hist}. You might ask why representations take place in such a huge and complex space when only a fraction of its representational power is actually used. I_{hist} should not be understood as *the* representational space, but *a* representational space, a very basic one. Any information the agent can acquire is representable as an event history $\eta \in I_{hist}$. Furthermore, I_{hist} is, in contrast to state space X, both well defined and completely task invariant and is as such very suitable for learning purposes. However, in many other respects I_{hist} is not the best representational space. I_{hist} contains a lot of redundant information, making it difficult to extract features relevant to the specific task. For this reason, a new ***derived information space*** I_{der} may be created. I_{der} should be seen as a simplification of I_{hist}, where relevant features are represented, while irrelevant information is not contained, [53, p.571]. The observant reader may think this sounds disturbingly similar to the formulation of state space. This observation is highly relevant and reflects to some extent the purpose of inferring I_{der}. The use of derived information spaces as bias in learning, and its relation to the state space, is further discussed in Sections 3.2 and 3.4.

2.4. Controller

The controller defined in Equation 1 can now be reformulated in a form that allows it to be used without full access to state space X :

$$u_k = \pi(\eta_k) \tag{7}$$

where $u_k \in U$ is the action vector issued at stage k and $\eta_k \in I_k$ is the agent's event history at stage k. π is defined here as a function from information history space to action space:

$$\pi : I_{hist} \rightarrow U. \tag{8}$$

In simple cases, a controller can be modeled as a function of only the most recent sensory-motor event. Systems based purely on such single-event controllers are called ***reactive systems*** [21]. Formally, these systems implement π as

$$u_k = \pi(y_k) \tag{9}$$

which can be seen as a special case of Equation 7. This definition of π is similar to Arkin's ***behavior mapping*** $\beta : S \rightarrow R$, where S and R are stimulus and response, respectively [5]. However, in the general case we use the definition of π given in Equation 7.

2.5. Behavior

The word behavior is commonly understood as an agent's actions in relation to the environment, but in the robotics community it has many different meanings. In the present work, behavior is understood as a purposeful way of acting. This does not imply that behaviors include explicit representations of goals, but from an observer's point of view, the behavior can be said to implement some kind of purpose, or goal. This argument is developed in Section 3.3.

Using the introduced terminology, a ***behavior*** B is defined as a subset of information history space $B \subset I_{hist}$. Each element in B is an event history η that represents one instance of the desired behavior.

Often, no explicit distinction is made between the observable interactions with the world, and the mechanisms producing these interactions. However, B describes nothing about how the behavior is produced, and therefore this notion of behavior is different than the terminology commonly used within behavior-based robot architectures [5, 27, 58, 68]. B is purely an intrinsic definition and describes exclusively the behavior from the agent's perspective.

3. Learning From Demonstration

Learning From Demonstration (LFD) is a well established technique for robot learning. An overview of early work is found in the work by Bakker and Kuniyoshi [6] while recent work and classification of the field is found in the survey by Argall et al. [4]. Another excellent survey of the area can be found in a recent book by Billard et al. [12]. The basic idea in LFD is that the robot learns to do things by observing other agents, be it human beings or other robots. Several flavors of this approach exist and the terminology used differs somewhat in published research. Similar approaches are presented under terms like *Imitation Learning*, *Learning From Experience*, *Learning From Observation* and *Robot Programming by Demonstration*. See the work by Argall et al. [4] for more details on terminology.

Research on LFD has been divided into four key problems: what, how, when and who to imitate [11, 12]. ***What to imitate*** refers to the problem of identifying which aspects of the demonstration are relevant for the task [20]. ***How to imitate*** is the question of how the skill is to be encoded. A central part of this issue is the ***correspondence problem*** [66, 67] which refers to the process of mapping the observed sequence of events to corresponding actions of the pupil. In most practical situations the pupil is not given an explicit set of demonstrations, but the pupil must detect when the teacher is doing something related to the task to be learned. This problem is known as ***when to imitate***. Finally, ***who to imitate*** refers to the identification of the teacher, which is also a difficult issue in many applications. These four questions are very general and can also be applied to learning situations with human or animal pupils. In practice, what and how to imitate are the most frequently studied problems within LFD.

New behavior can be demonstrated to a robot in many ways, for example by having the robot pupil watch the teacher demonstrate the desired behavior. Here we focus on LFD where the teacher directly controls the robot, e.g. by teleoperation. The recorded data sequence from such a control session, including both executed motor commands and sensor readings, is denoted ***demonstration***. The purpose of LFD is to create a controller π capable of reproducing the demonstrated behavior. While there are many other ways to demonstrate a new behavior to a robot, LFD via teleoperation constitutes a well defined setting that can be generalized to many practical applications. Formally, a demonstration is, in this setting, an event history $\eta_k \in I_{hist}$ (refer to Equation 4) where $\tilde{u}_{k-1}$ is the sequence of actions issued by the teacher up to stage $k-1$ and $\tilde{y}_k$ is the sequence of observations up to stage k.

In this setting, a direct correspondence between recorded events in a demonstration and sensors and actuators is assumed (a direct record mapping and no embodiment mapping, following the terminology by Argall et al. [4]). The observations y_k in the demonstration are assumed to correspond to the observations that are generated in real-time by the sensors and sent to the controller. Furthermore, the observed action variables u_k are assumed to directly correspond to the actuator signals generated by the controller π. This relates to *self-*

imitation, i.e., the pupil learns by performing the actions itself, with help from a teacher [78, 79]. Self-imitation, in contrast to imitation of others, avoids two difficult problems. Firstly, the problem of observing the teacher's actions, and secondly, the correspondence problem.

LFD has its roots in the more general approach to create computer programs from demonstrations, known as *Programming By Demonstration (PBD)* or *Programming By Example (PBE)*, e.g. [26, 54]. However, modern LFD is far from these general approaches. This paper presents a formalism for robot learning through demonstration, which, while it can be seen as the creation of a specific kind of computer programs, does not aim at the wider interpretations of PBD or PBE.

The goal of LFD is, in this context, to generate a controller π that enables a robot to *repeat* a demonstrated behavior B. π may be a state-action mapping, a model of the world dynamics (system model) or a model of action pre- and postconditions (plans), see the work by Argall et al. [4] for details. If successful, the robot is said to have learned behavior B. Formally, the process of learning B from a set of N demonstrations b is understood as selecting π from the *controller space* Π using a *learning function* λ:

$$\pi = \lambda(b) \in \Pi \tag{10}$$

where b is the set of event histories η that constitute the demonstration. The LFD process is illustrated in Figure 1. Π contains all possible controllers for a specific chosen observation space and action space. This is of course a huge space that is never computed explicitly.

The selected controller π must have specific qualities for the learning to be regarded successful. These qualities are related to the event histories η that may be generated by a robot using controller π. The *realization space* $R \subset I_{hist}$ for a π is defined as the set of all such event histories, generated by the *realization function* Λ:

$$R = \Lambda(\pi) \in I_{hist} \tag{11}$$

Λ can be seen as an abstraction of the physical robot placed in a particular environment and controlled by a specific π, able to produce the set of all possible trajectories through I_{hist}. Of course, the robot can not control the produced event histories $\eta \in R$ entirely on its own, but relies on an external component, the environment. This creates a nice analogy to λ, which also relies on an external component, called *bias*. Thus the learning function λ can be seen as the inverse function of the robot represented by Λ. λ maps a set of event histories to a controller and Λ maps a controller to a set of event histories. This is further developed in Section 3.2.

The process of selecting π has many similarities to system identification, where a model of the system is constructed from observed input and output data [55]. The system, consisting of the agent and its environment, is modeled such that the system output u_{k+1} can be predicted given a sequence of previous inputs and outputs η_k until stage k. However, the aim of system identification is in one sense much more ambitious than LFD, since the system's response to any input y_k is to be predicted. In LFD, we are satisfied with a π producing an action that, if possible, leads to an event sequence $\eta_{k+1} \in B$ given that $\eta_k \in B$. In other words, LFD does not necessarily model the outcome of all possible actions u_k in each state, only the ones that occur for the robot in a particular environment.

B should be understood as the set of event histories the human teacher associates with a particular desired behavior. For example, if the teacher wants to teach the robot to move to a door, B would contain all event histories where the robot ends up by a door, in an acceptable way. The behavior must be formulated such that the robot is able to reproduce the behavior in all desired environments. There may be situations in which the robot can not distinguish between significant aspects of the world. In these cases, the robot's sensing capabilities or other aspects of the behavior have to be modified. Assume that the *move-to-door* behavior is to be applied to a robot in a hotel environment. The robot must now be able to separate between doors. One alternative is to add a new sensor allowing the robot to directly identify each door it approaches, resulting in a redefined I_{hist}. Another alternative is to change the behavior such that the robot can use existing sensors, e.g. wheel odometry, in order to distinguish different doors by their locations. This corresponds to a modification of B.

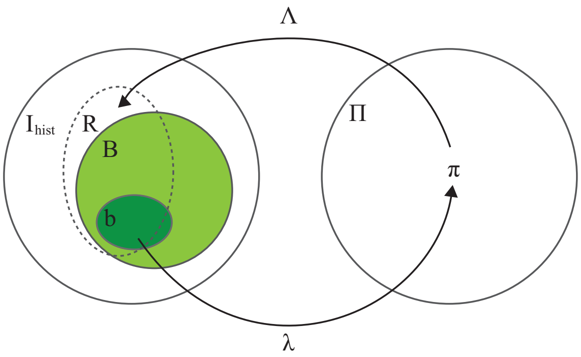

Figure 1. *The LFD process. The light-colored area represents the wanted behavior B which is demonstrated with N training demonstrations $b = \{\eta^{(1)}, ..., \eta^{(N)}\} \subset B$ represented by the dark-colored area. The learning function λ creates a controller $\pi \in \Pi$. In interaction with the environment, π realizes (repeats) the learned behavior. The realization set $R \subset I_{hist}$ is marked by the dashed line.*

The quality of the generated π is typically described as the ability to "repeat a behavior", which is the topic of the next section.

3.1. What does it mean to repeat a behavior?

The goal of LFD is to generate a controller π that enables a robot to *repeat* a demonstrated behavior B given a set of demonstrations b. This may sound like a well defined mission, but is actually both vague and ambiguous. Consider the following example of a seemingly trivial demonstration.

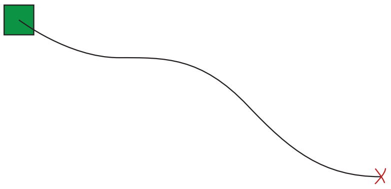

Figure 2. *A simple demonstration where the tip of a robot arm starts at the red cross in the lower right corner and moves over the table until it is positioned over the green cube. The demonstration can be interpreted in a number of fundamentally different ways.*

Observe a sequence of sensory-motor events describing a robot arm moving over a table, finally stopping when positioned above a green cube (Figure 2). What does it mean to repeat this sequence of events?

One could imagine a vast number of interpretations. Here are a few examples.

1. Assuming that the path is the important aspect of the demonstration, a successful controller may be written as $u = \pi_{PATH}(y)$ where the function π_{PATH} computes an action u for each pose y, such that the arm follows the demonstrated path. This kind of learning scenario refers to traditional programming of industrial robot arms, as well as path-tracking autonomous vehicles, e.g. [43].

2. Instead, if the demonstration is seen as an example of how to reach the final position, the path itself becomes irrelevant and the controller described above would not be suitable. In this case, a successful controller could be written as $u = \pi_{TARGET}(y)$ where the function π_{TARGET} uses inverse kinematics to compute actions such that the tip of the robot arm reaches the target.

Case 1 corresponds to what is often called *action-level imitation* [22] where the robot carries out the same actions as the demonstrator. Case 2 is often called *functional imitation* [29] in which the robot is supposed to achieve the same effect on the environment [67]. In the work by Alissandrakis et al. [2], the quality of action-level imitation is measured in state and action metrics while functional imitation is measured in effect metrics. State and action metrics define the similarity of behaviors in terms of the state and/or actions of the agent, while effect metrics define behavior in terms of their effect on the environment.

Within these two categories one could imagine a vast number of interpretations. Should the observed sequence of positions be understood as fixed coordinates, or relative to the robot arm's starting position? Is the green cube really the relevant target, or is the target defined by an absolute position? Is the target a cube of any color, or or is the target perhaps any green object? Using many demonstrations of the same behavior reduces some of the ambiguity, but in general it is impossible for the learner to tell which interpretation is "correct" without further information. In fact, the learner can not even enumerate a set of possible interpretations without a specification of state variables relevant for the task to be learned. The discussion about what it means to repeat a behavior becomes complicated further when the robot acts in a dynamic, non-deterministic and partially accessible [77, ch.2] environment. Demonstrated event sequences may be both incomplete and contain mistakes that should not be learned or repeated [28].

If the robot manages to successfully repeat a demonstrated behavior under different conditions than during the demonstration we say that the robot is able to *generalize* the demonstrated behavior. More specifically, we refer to the robot's ability to produce an event history $\eta_k \in B$, under conditions η_{k-1} not identical to the ones appearing during the demonstrations in b. This can be formally described as how well the realization space R corresponds to the desired behavior B, e.g. as a minimization of $R \setminus B$ and $B \setminus R$ (refer to Figure 1).

Generalization can also be viewed as an extension of b by interpolation or extrapolation of the demonstrated event histories. For this to work one has to specify the aspects of the demonstrated data that are important, i.e., the previously mentioned problem of *what to imitate* (Section 3). One approach is to introduce a *metric of imitation performance* [1, 2, 10]. Repeating a demonstration means minimizing the distance between the demonstrations and the repetitions using this metric. To find the metric, the variability in many demonstrations is exploited such that the essential components of the task can be extracted. One promising approach to construct such a metric is to use the demonstrations to impose constraints in a dynamical system [24, 38, 44]. Giovannangeli and Gaussier [35] use human-robot interaction to improve generalization when learning sensory-motor behaviors for homing and path following. In the described work, teaching by error correction (proscriptive learning), is shown to give superior generalization compared to a regular demonstration (prescriptive learning). The generalization problem is also acknowledged outside the LFD community. In *Machine Learning*, the term *generalization performance* of a learning algorithm relates to "its prediction capability on independent test data" [41, p.193] which is identical to the common usage of the term in robotics. The general problem with machine learning in high-dimensional spaces is often expressed as the curse of dimensionality [33, p.170], and is highly relevant also for robots with high-dimensional observation and action spaces. Learning in such situations becomes inherently difficult since the demonstrated data fills history information space very sparsely and interpolation and extrapolation become highly risky operations. The situation is related to the *No Free Lunch Theorem* [85], which states that for a large class of machine learning algorithms, there is no universal best algorithm to solve all problems. Instead, an algorithm has to be specialized to the problem under consideration to guarantee its superiority over any random algorithm. This specialization consists of additional task-dependent information that has to be supplied to the learning algorithm as bias. In the case of LFD, possible sources of bias are the robot's prior knowledge, feedback from the environment when the robot tries to repeat the demonstrated behavior and human feedback before, during, and after learning. The bias concept is further investigated in the next section.

3.2. Bias

The bias of a machine learning algorithm is defined as "any basis for choosing one generalization over another, other than strict consistency with the observed training instances" [63]. The basis may be seen as form of pre-evidential judgment, or prejudice regarding the structure of the data or the data generating process. In the case of numerical regression, assuming a linear relation between input and output corresponds to a high bias, while a cubic model corresponds to a lower bias. In the case of LFD, bias can be applied to three different parts of the problem definition:

1. Sensor variables. This can involve selection of relevant sensors, or extraction of specific features that are judged relevant for the specific task. It may also involve creation of intelligent sensors to facilitate feature extraction.

2. Action variables. Most often this involves restricting the output of the controller π to one or a few actuators. For example when learning a grip operation, the actions for moving the robot may be regarded irrelevant while the gripper motion is highly relevant. This reduces the size of action space.

3. Controller function π. Bias can restrict the functional form of π, e.g. to an artificial neural network of a specific size and architecture. Bias can also be expressed as general requirements of π, such as smoothness criterion or lower/upper bounds. The use of predefined skills as described below is another example.

Bias can be introduced into the learning process in a number of ways. First of all, it may be hard-coded into the learning algorithm, e.g. by choosing a specific neural network [57] or rule based framework Hellström [42] to represent π. Another common and very powerful technique to introduce bias is to use predefined skills or behavior primitives. Besides being biologically motivated [36, 64], the technique is commonly used in robotics research, e.g. [34, 59, 61, 68]. Learning is in this case reduced to selection of the right primitives and parameter estimation to adjust the primitives to the demonstrated data. The introduction of primitives is a way to reduce the dimensionality of the learning problem (i.e. to deal with the ***curse of dimensionality*** mentioned above). The set of primitives is obviously much smaller than Π which clearly simplifies learning. An analogy is numerical regression with a large feed-forward neural network compared to a low-level polynomial. The polynomial introduces bias that makes learning much easier, at the price of limiting the solution to the specific functional form of the bias.

Regarding bias for sensors and actuators, it is common to hard-code a set of relevant sensors and action variables for the task at hand, or to pre-process the data before feeding it to the learning algorithm. This kind of bias may also be introduced by interaction with the human teacher who tells the robot to use specific sensor modalities. Saunders and coworkers present an approach where relevant elements of the state vector are weighted based on their information gain and on manual selection from a teacher [70, 79].

Bias may also be subject to meta learning, suitable sensors can for example be selected based on demonstrated data. This relates to ***attention*** and ***saliency*** which are important concepts in theories for human and animal learning. The term ***shared attention*** refers to a teacher's and a learner's simultaneous attention to the same objects. Scassellati used the Cog platform [80] to investigate shared attention between humans and robots. Saliency refers to the components of the environment that are important for a given task, and it clearly introduces a bias by reducing the size of observation space Y. Breazeal and Scassellati, [18] describe the relationship between attention and saliency and how the concepts can be used to facilitate learning in robotics.

These techniques relate to the psychological term ***scaffolding***, which is used to denote interaction between caretakers and infants in order to reduce distractions, marking a task's important attributes and reducing the number of degrees of freedom in the learning task in general [19, 87]. All these operations aim at simplifying the learning task by introducing bias to the problem definition.

From a formal perspective, bias regarding sensor and action variables may be introduced by moving away from I_{hist} into a new, derived information space I_{der} [53, p.571]. I_{der} is a reformulated or pre-processed version of the information in I_{hist}. The mapping from I_{hist} to I_{der} is denoted κ, and may have an arbitrary shape:

$$\kappa : I_{hist} \rightarrow I_{der}. \tag{12}$$

An element of I_{der} is referred to as a ***derived event history*** η_{der} and can be generated from $\eta \in I_{hist}$ using the mapping κ. Therefore, I_{der} does not serve as a general purpose representational space as I_{hist} does, but rather as a task-specific representation where relevant features become salient, while irrelevant information is not retained. The purpose of I_{der} is similar to the purpose of the state space X. In fact, a state space is one possible instance of I_{der}, but there are numerous other possible derived information spaces that do not aim at representing states in the world.

The LFD process with bias included is illustrated in Figure 3. Various ways to introduce bias regarding the control function π result in a reduced set $\Pi_p \subset \Pi$. The learning function λ maps from the derived information space I_{der} instead of straight from I_{hist}. This extended formulation of LFD is further discussed in Section 3.4.

Referring to Figure 3, the ***what to imitate*** question shows up as a transformation problem from I_{hist} to I_{der}, i.e., an identification of the relevant aspects of the task. Since we are focusing on a self-imitation setting, the correspondence problem is not present here. However, there is still the problem of selecting a controller $\pi_p \subseteq \Pi_p$ based on b_{der}, reflecting the remaining parts of the how to imitate question. When to imitate appears as ensuring that $b \subseteq B$, i.e., that everything in the demonstration set b is actually part of the desired behavior.

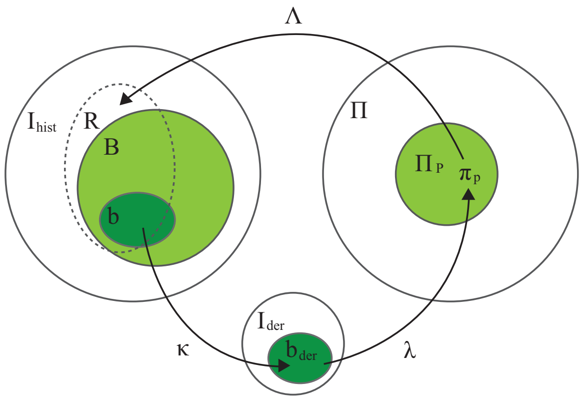

Figure 3. The LFD process with bias introduced. A derived information space I_{der} is introduced as a space where the behavior may be represented in a task-specific way. Training data b is mapped into I_{der} with an information mapping κ. The pre-processed information in I_{der} and various ways to introduce bias in λ result in a reduced set of possible controllers Π_P, illustrated by the light colored area in Π. Compare with Figure 1.

Our discussion about bias has so far been focused on knowledge intentionally introduced into the system to facilitate learning. We like to refer to this kind of information as ***ontological bias***. However, there are also a vast number of restrictions to the problem introduced for other reasons. As mentioned before, selecting a specific type of algorithm to represent π will introduce bias. A particular configuration of the robot's sensors and actuators restricts the ways in which it can solve a particular task. Often the choice of physical platform and software architecture is made for practical reasons rather than for an understanding of ontological implications. We like to phrase these kind of restrictions as ***pragmatical bias***.

Independent of the type of bias being introduced into the system, it limits the behaviors the robot can learn. Consequently bias is not necessarily positive. Instead, one should aim at a suitable level of bias, such that the robot can learn as many interesting behaviors as possible, while still being able to generalize correctly.

As mentioned above, using pre-defined skills or behavior primitives is a common way to define Π_p. The demonstrated data are in such cases used to identify a suitable primitive and may also be used to set parameters for the selected primitive. One way to define such primitives is to associate them with achievement of specific ***goals***. This concept deserves special attention and is analyzed further in the next section.

3.3. Goal

The success or failure to repeat the demonstrated behavior is most often judged by the human demonstrator, and to describe the human intentions we use the word ***goal***. The goal of a behavior is a human concept and can be of two major types [68]:

1. Maintenance goals. A specific condition has to be maintained for a time interval, such as the path-tracking scenario described in Example 1 in Section 3.1.

2. Achievement goals. A specific condition has to be reached, such as the motion to a green cube in Example 2 in Section 3.1.

A behavior B was earlier introduced as a set of event histories that, from a teacher's perspective, fulfills some common purpose. This can be understood as after performing B, specific conditions in the world are satisfied. This is analogous with the common goal formulation from classical AI, where a goal G is a set of states in state space [77]:

$$G \subset X. \tag{13}$$

All the information the agent acquires about G is accumulated over time in $\tilde{y}$ and $\tilde{u}$. Therefore, any goal G which can be measured with the agent's sensors can also be formulated as a set of event histories $\eta \in I_{hist}$:

$$G_I \subset I_{hist}. \tag{14}$$

This should be understood as after observing an $\eta \in G_I$ we know that G is satisfied. A consequence of this formulation is that behaviors and goals are represented in the same way, and since any $\eta \in B$ by definition satisfies the goal of B, G_I and B become identical:

$$G_I = B. \tag{15}$$

This may also be explained from the reversed perspective. When X is viewed as a derived information space, G will cast a pre-image into I_{hist} which per definition will be identical to B. Still, this formulation of goals is not very satisfying. In state space, G most often has an intentional definition, a neat formulation that describes the minimum requirements. However, in the task invariant I_{hist}, a neat goal can not be formulated since no bias has been introduced.

When a human teacher speaks about goals he or she uses task specific information which in principle could be transferred to the robot as bias. This is partly what is done when a state space is defined in classical AI. But the information a human uses to formulate goals may not be necessary for executing the same acts, maybe not even helpful. This argument is nicely illustrated in the *frame of reference* [14, 73]. By assuming the necessity for a human goal formulation we impose our own frame of reference upon the agent, and may make representation of the behavior much more complicated than it may be from the agent's perspective.

A common way to introduce this separation between the human's and the robot's frame of reference is to introduce pre-programmed primitives. The set of known primitives creates a space where the human teacher can easily get an understanding of what the robot is doing, while the specific controllers can create local information spaces suitable for the specific primitive. The use of primitives is further developed in the following section.

3.4. Learning with behavior primitives

Based on the concepts of behavior, bias, and goal introduced above, the learning task defined in Equation 10 is here refined. In Section 3.1 it was concluded that λ requires some bias to be able to find a suitable controller, as illustrated in Figure 3. In the most basic form of LFD, λ is simply learned by fitting the demonstrated data to a more or less general functional form, such as a neural network [57] or a rule base framework [42] which in such cases represents the reduced controller set Π_P in Figure 3. The use of primitives, which was introduced in Section 2.1, is fully compatible with this description of learning bias such that learning consists of matching a demonstration with a predefined primitive. This process is denoted *behavior recognition* and can be approached in a number of ways as described below.

The description of LFD given above is valid for demonstrations of behaviors that can be repeated by choosing one single primitive. More complex behaviors demand sequences or combinations of primitives. For a given robot and class of learning scenarios, the set of primitives Π_P is normally chosen such that a demonstration may be divided into segments where each segment can be repeated by choosing the right primitive. The general LFD process illustrated in Figure 3 is here extended to include handling of such sequences. Some types of behaviors are better described as combinations of several primitives executed in parallel, e.g. [69]. This organization is common in behavior-based architectures, e.g. [27, 58]. However, recognition of primitives executed in parallel is incredibly complex in the general case. Furthermore, these systems require a coordination function that integrate motor commands from parallel primitives. Due to these issues, parallel primitives are less common in LFD applications and we have therefore chosen to focus on the purely sequential case.

Let us first look from a post learning perspective at how sequence control can be described for a robot using a set Π_P of predefined primitives π_p. To include the assignment of parameters for parameterized primitives into the learning, Π_P is in the following regarded as containing all possible parameterizations of primitives. Control can now be divided into two steps:

1. Action selection where a function π_{sel} selects a primitive $\pi_p \in \Pi_P$:

   $$\pi_p = \pi_{sel}(\eta_{der}) \tag{16}$$

   where π_{sel} performs the mapping

   $$\pi_{sel} : I_{der} \rightarrow \Pi_P \tag{17}$$

   $\eta_{der} \in I_{der}$ is a pre-processed or derived version of the original event history $\eta \in I_{hist}$, constructed by an information mapping function κ [53, p.571], defined in Equation 12.

2. Low-level control using the chosen controller π_p to generate an action u_k.

Stepping back to the learning phase, the problem is now reduced to finding the action selection function π_{sel} using demonstrated data b pre-processed with the information mapping κ into the derived information space I_{der} (see Figure 4)[1]. In this way, the dimensionality of the learning problem is drastically reduced since λ is now selecting suitable $\pi_{sel} \in \Pi_{sel}$ based on the pre-processed trajectory information in I_{der} rather than working on the full I_{hist} and Π spaces. Compare with Figures 1 and 3.

While the approaches to sequence learning with primitives vary widely, the process of finding π_{sel} can be divided into three tasks:

1. *Behavior segmentation* where a demonstration $\eta^{(i)}$ is divided into smaller segments, referred to as *task segments*.

2. *Behavior recognition* where each segment is associated with a primitive $\pi_p \in \Pi_P$.

[1] *By comparing Equations 16 and 17 with Equations 7 and 8, the primitives π_p may be seen as generalized actions, generated by a controller π_{sel}. Another interesting analogy can be made between action selection and the correspondence problem, i.e., the problem of finding the action(s) that corresponds to an observed event sequence. Viewing the primitives as actions leads to an equivalent problem formulation for action selection; find the primitive that corresponds to an observed event sequence.*

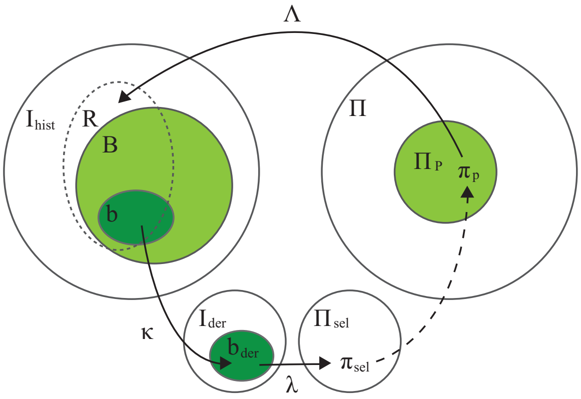

Figure 4. An extended version of the LFD process illustrated in Figure 3. Bias is here introduced into the learning process by restricting Π to a set of primitives Π_P. Primitives π_p are selected by selection function $\pi_{sel} : I_{der} \rightarrow \Pi_P$. Solid lines represent function mappings while the dashed line represents the evaluation of π_{sel}.

3. *Behavior coordination*, referring to identification of rules or switching conditions for how the primitives are to be combined.

Referring to Figure 4, these tasks are realized by the function λ. In practice, task 1 and 2 are often intertwined. For Task 1, several approaches exist, for example variance thresholding [46, 51], repeated pattern correlation [49, 75, 76], thresholding mean velocity of joints [34, 65] and entropy-based segmentation [25]. Auto-associative neural networks have also been used for segmentation, both by measuring network reconstruction performance [15] and by identifying bifurcations in the network attractor dynamics [49, 50]. Calinon and coworkers [24] used Dynamic Time Warping in combination with Gaussian Mixture Regression to decompose movement trajectories of a humanoid robot.

Task 2 is commonly seen as a classification problem. For example, Bentivegna [8] uses a nearest-neighbor classifier on state data to identify skills in a marble maze and an air hockey game. In both these setups, each primitive is assigned a query point in state space, which is compared with the current system state. Pook and Ballard [74] present an approach where sliding windows of data are classified using Learning Vector Quantization in combination with a k-NN classifier. The complexity of the distance measure is highly dependent on the complexity of B. For simple behaviors, a Euclidean distance function has been shown to work well [9]. However, for more complex behaviors, other measures are necessary. Tani [83, 84] does both recognition of behavior primitives and segmentation with extended recurrent neural networks that model different behavior primitives depending on the parametric bias in the network model. Recognition is done by finding the optimal parametric bias for an observed sensory-motor sequence. Calinon and colleagues use Hidden Markov Models in combination with Principal Component Analysis to compute the likelihood that the observed data was generated by the model [23, 24].

One approach that addresses the complexity of higher level primitives can be found in work by Nicolescu [68], where two behaviors are regarded as being similar if their respective preconditions and goals match, regardless of their internal differences. Nicolescu utilizes the postconditions to recognize primitives in demonstrated data, i.e., task 1 and 2 as described above. Recognized primitives are arranged in a behavior network and during execution the behaviors' preconditions in combination with the network links are used for behavior coordination, Task 3. Formally, any sequence of recognized primitives can be seen as an element in a derived information space I_{der}, and consequently a behavior network, represented as a set of behavior sequences, constitutes a subspace of that I_{der}. In this setting, the definition of post-conditions for each primitive constitutes an information mapping κ from I_{hist} to I_{der} and the preconditions take part in the implementation of the coordination function π_{sel}. The primitive controller itself is represented by $\pi_p \in \Pi_p$. Compare with Figure 4.

Demiris and Johnson [31] present a different approach where all primitive controllers are continuously running in parallel, predicting actions in response to incoming sensor data. The prediction errors are then used to estimate how well each primitive represents the demonstrated behavior. This approach is similar to our own method β*-comparison*, which is also used for some primitives in the present example, c.f., Section 4. Even though theoretically appealing and with strong connections to biological findings, see [31] for details, direct comparison of predicted actions become infeasible for complex primitives. The method presented by Demiris and Johnson, as well as our β*-comparison*, has problems capturing the similarity of behaviors that may be executed in many different ways, leading to the same goal. One way to handle these issues is to move from a direct comparison of actions in I_{hist} to more abstract concepts of actions or events in a derived event history $\eta_{der} \in I_{der}$. An evaluation of β*-comparison* and two other methods for behavior recognition can be found in [15]. In a generalized sense these methods should be seen as an attempt to create a *metric of imitation performance*, as discussed in Section 3.1.

Sometimes, a demonstrated behavior can not be decomposed into a sequence of known discrete primitives. Several metrics may conflict and cause ambiguities in behavior recognition. In these situations, continuous task representations are preferable since they can better describe a smooth transition from one metric to another, see for instance [24].

A distributed approach to Task 3 is presented by Maes and Brooks [56]. Global feedback is used, allowing the primitives themselves to learn suitable activation conditions by correlating particular stimuli with positive or negative feedback. The feedback functions in combination with the primitives themselves constitute the coordination function π_{sel}. Another approach to behavior coordination is found in the MOSAIC architecture [39, 40, 86]. MOSAIC utilizes forward models paired with primitive controllers. Each forward model computes a responsibility signal as a measure of how well the paired controller can handle the present situation. When combined with a responsibility predictor this architecture forms a powerful coordination system. MOSAIC is a theoretical framework but the HAMMER architecture [30, 32], which has been implemented and tested on robots, captures many aspects of MOSAIC. Both these architectures are put in relation to LFD in our own recent work [13]. A key aspect of this approach is the pairing of forward models (predictors) and inverse models (controllers) in a model-free way. We are analyzing this issue deeper and propose a possible solution based on the algorithm Predictive Sequence Learning algorithm in other recent publications [16, 17].

There are several approaches to identify relevant aspects of the task that do not employ behavior primitives. While we limit the present review to approaches using primitives, the work by Kulič et al. [52] is worth mentioning even though it does not directly apply behavior primitives. In this approach, demonstrations of movement patterns are encoded in Hidden Markov Models and then clustered into groups using Hierarchical Agglomerative Clustering. Groups are formed incrementally as new demonstrations are added, which makes this approach display many of the advantages with behavior primitives as described here. Furthermore, Kulič et al. put forward the advantage of a hierarchical organization of behavior, a claim we support strongly and discuss deeper in other work [13].

Adding to the motivations presented above, one important reason for the use of primitives in LFD is that primitives constitute high level representations of the demonstrated behavior. Primitives can be labeled in meaningful ways, which helps establish a common understanding between the human teacher and the robot pupil. It is natural for humans to break down sequences of actions into meaningful sections and adults appear to agree upon how segmentation should be made [7]. We therefore believe that identification and recombination of behavior primitives is a critical aspect of LFD.

4. Demonstrator

The concepts and theory introduced above are here illustrated with an experiment in which a Khepera robot [48] is used in an LFD setting. This experimental setup is on purpose simplified to illustrate how ambiguous even a very simple demonstration may be, and how the proposed formalism can be used to describe the LFD process.

The Khepera robot has eight infra-red proximity sensors mounted around the rim of the robot. The limited sensing capabilities have for this experiment been augmented by an external camera mounted above the robot arena. The setup can be seen in Figure 5 and an example image from the top mounted camera can be seen in Figure 6. The robot is equipped with a gripper and is placed in an environment with a number of wood blocks and two colored areas located in one side of the scene.

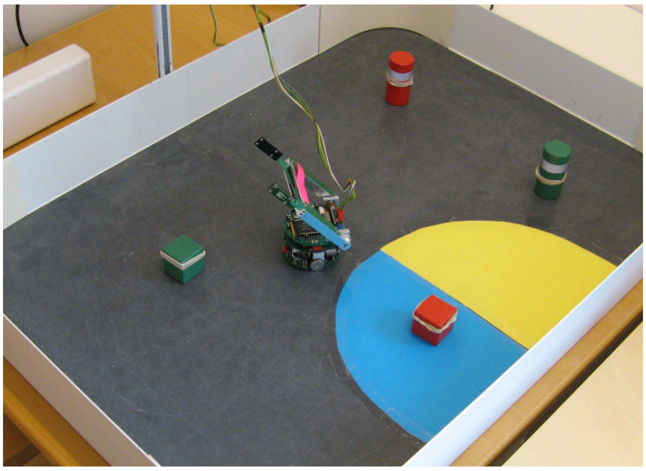

Figure 5. Experimental setup. In the center is a Khepera robot [48] with a gripper that can be raised and lowered. The objects around the scene are painted wood blocks. Rubber bands have been placed around the objects to facilitate gripping. A camera has been mounted directly above the scene, see Figure 6.

The experiment comprises a sequence learning task in which a human intends to teach a robot to pick up cubes and place them in the blue-colored corner area. To demonstrate the wanted behavior, the human tele-operates the robot towards a red cube, grips it, lifts it, moves to the blue area and drops down the cube. The robot should then be able to repeat the demonstrated behavior. The reader is referred to Figure 1 which summarizes much of the discussed formalism.

Observation space Y comprises the camera image (Figure 6), data from the eight proximity sensors, position sensors for gripper and gripper arm and an optical barrier detecting objects in the gripper. *Action space* U comprises the speed of the left and right wheel, and the speeds of the motors controlling gripper lift motion and gripper close motion. Sequences $\tilde{y}_k$ and $\tilde{u}_k$ (Equations 2 and 3) are combined into *history information states* $\eta_k \in I_{hist}$ (Equations 5 and 6). I_{hist} is a huge space comprising all possible sensor and action sequences the robot in principle can experience. Given the task at hand, a more suitable *derived information space* I_{der} is defined. It comprises sequences of the following entities derived from Y and U: Object properties *distance, direction, orientation, type*, and *color* where *type* is either *cube* or *cylinder*. Directions and orientations are given in a coordinate system relative to the robot. Distance and direction to the centroids of the two colored areas are also extracted. Technically, these entities are extracted from the camera image using a combination of image analysis tools, including color segmentation, Sobel edge detection, Hough transform and mathematics morphology. Formally, these techniques are parts of the κ, defined in Equation 12.

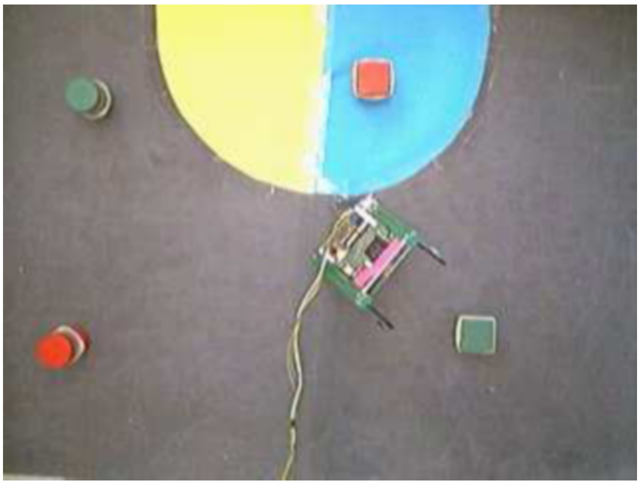

Figure 6. Example image from top mounted camera. A pink tape has been placed on the Khepera gripper to facilitate recognition of the robot's position and orientation.

The generation of I_{der} should be seen as the first of many kinds of *biases* that we introduce in order to make the learning task feasible. This bias depends on the available sensors and actuators and also on the task at hand. It is clear that the dimensionality of the learning problem is significantly reduced by replacing the camera image in I_{hist} by a small number of object properties.

The demonstrator performs the wanted task by tele-operating the robot as described above. The resulting recorded data $b_{der} \subset I_{der}$ is a set of event histories constituting the input to the learning function λ. To support this process, the universe Π of all possible controllers is reduced to a much smaller set Π_p that comprises pre-defined high-level behavior primitives. The following primitives are defined: *move_to_object*, *move_to_area, grip, release, lift* and *put_down*. The *move_to_object* primitive takes two parameters *color* and *type,* where $color = C \subseteq \{red, green, blue, yellow\}$ and $type = T \subseteq \{cube, cylinder\}$. The *move_to_area* primitive takes one argument *color* just like *move_to_object*, but does not have any *type* parameter. One could of course imagine many other possible parameters for the these primitives, e.g. position and size, but the included parameters suffice for the present example. Referring to Section 3.3, each parametrization of the *move_to_object* and *move_to_area* primitives is associated with a specific goal G_l (Equation 14). As been already mentioned, this is a very efficient way of introducing additional bias in learning such that complex behaviors can be learned by few or even a single demonstration. Conceptually, Π_p comprises all possible parameterizations of the primitives.

To keep the example simple, all primitives are hard-coded into the robot, i.e., both I_{der} and Π_p are defined manually. However, in a realistic setting primitives are often created during a previous learning phase, as has been shown in for example the work by Saunders et al. [70, 79]. The use of primitives should be seen as a way to reuse knowledge that may come either from a programmer manually designing the primitive, or from a previous learning phase. Formally, this is described as a gradual redefinition of I_{der} and Π_p which corresponds to the concept of scaffolding described above.

To learn a sequence of these primitives, the three steps described in Section 3.4 are performed. Behavior segmentation and recognition are executed in one step by continuously matching each primitive against b_{der}. The recognition method differs between different primitives. For *grip, release, lift* and *put_down*, the start and end positions of the gripper are used to *indicate that* the corresponding primitive has been executed. For *move_to_object* and *move_to_area* the behavior recognition method $\beta - comparison$ [15] is used. In this approach, an action vector for each parameterized primitive is computed, creating a set of hypothesis. Each action vector is then compared to the observed b_{der} creating an error measure for each hypothesis. If the error remains low while the robot is approaching a specific target object, the hypotheses is confirmed and a *move_to* primitive with the corresponding parametrization can be inserted into the recognized sequence.

Each primitive specifies a set of finish conditions, e.g. that the robot should be within gripping range of a target object for *move_to_object* to complete. The end result of the learning process λ is a function $\pi_{sel} \in \Pi_{sel}$ that selects an appropriate primitive $\pi_p \in \Pi_p$ given the current event history η_{der} (Equation 16). In this way, π_{sel} acts as a sequencer and the actual control of the robot and gripper motion is done by the currently selected primitive π_p.

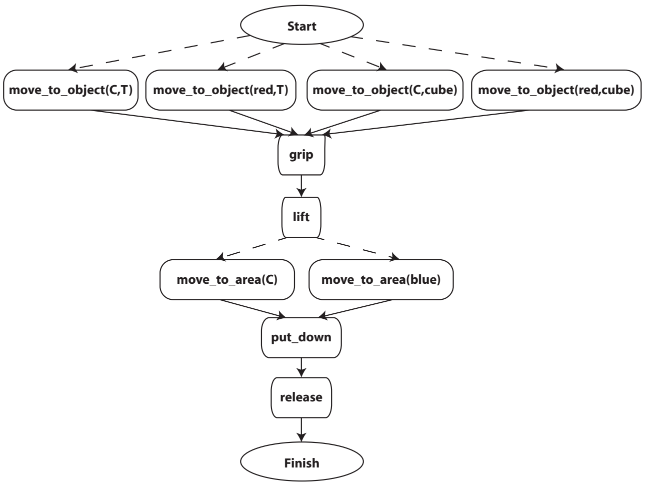

Figure 7. A schematic of the demonstrated sequence going vertically from top to bottom, where each square represents the execution of a primitive. Alternative interpretations of the demonstrated sequence are drawn horizontally, with the most general interpretation to the left and the most specific to the right. C and T are unspecified attributes representing all, or a subset of, possible values for color and type, respectively. Dashed lines mark ambiguous steps in the sequence, that require further information.

Even with the bias introduced by the construction of I_{der} and by the pre-defined behavior primitives in Π_p a substantial uncertainty, similar to the one discussed in Section 3.1, remains. This is illustrated in Figure 7 where alternative interpretations at each step are drawn horizontally and time flows vertically from top to bottom. In the shown example, the first part of the demonstration may be interpreted in four ways; *move_to_object(C,T), move_to_object(red,T), move_to_object(C,cube), move_to(red,cube)*. The second and third primitives *grip* and *lift* are uniquely identified while *move_to_area* is subject to similar ambiguity as *move_to_object*. Finally, the primitives *put_down* and *release* are uniquely identified. The alternatives for each ambiguity represent generalizations along different feature axes. In Section 3.1 this is described as interpolation or extrapolation of the demonstrated event histories η. Figure 7 illustrates a subspace of Π_{sel}. With the ambiguities resolved, through human feedback or other kinds of bias, the resulting sequence represents an instance of $\pi_{sel} \in \Pi_{sel}$ as defined in Equation 17.

Various types of feedback from the human can be applied such that the ambiguous sequence collapses into a single well defined sequence of behavior primitives that will enable repetition of the demonstrated behavior according to the user's intentions. In the described experiment, the human teacher manually selects the appropriate alternatives in a dialog system such that the generated π_{sel} will execute the sequence ž*move_to_object(C,cube), grip, lift, move_to_area(blue), put_down, release*~. The robot is then able to repeat the intended sequence of primitives and autonomously move cubes of any color to the blue area.

To sum up, the present example demonstrates how the huge and complex I_{hist} can be transformed into a significantly smaller and more human interpretable space I_{der}. On the controller side, the set of all possible controllers Π have been reduced by introducing a set of primitives Π_p that can be composed into sequences by Π_{sel}, c.f. Figure 4. A more detailed description of the experimental setup will be presented in future work, including a graphical interface in which the human user is able to give feedback during and after a demonstration, in order to resolve ambiguities such as the one illustrated in Figure 7.

5. Summary

A formalism for robot behaviors and *Learning from Demonstration* (LFD) is presented. Building on terminology from LaValle [53, ch.11], an agent's sensory-motor history is conveniently described by an event history, and a controller maps event histories to actions in action space. As illustrated in Figure 1, a demonstration of a particular behavior can be seen as an event history $\eta \in b$, and the behavior itself as the large set B of allowed event histories, i.e., all possible ways to realize the desired behavior. The quality of the learned controller can be judged by the similarity between B and the realization space R.

The vague and ill-posed meaning of *repeating* a demonstrated behavior is discussed from a machine learning perspective. The concept of *generalization* is defined in the framework of event histories and leads to a discussion of bias in learning. In LFD, bias is essential and can be introduced before, during, and after demonstration as feedback from the human teacher. The huge information history space may be reduced to a derived space, suitable for a limited set of tasks. Behavior primitives are another common way to introduce bias, and are often associated with specific *goals*, which are explicitly or implicitly defined for each primitive. LFD can at a higher level be described as controller selection. In this context, learning consists of finding and tuning a suitable primitive. More complex behaviors can be created by combining several primitives into sequences. LFD can then be described in three steps, *behavior segmentation, behavior recognition* and *behavior coordination.*

When using primitives created during a previous learning phase, learning can be seen as an evolutionary process where new knowledge is gained through the use of previous knowledge as bias. Formally, this is described as a gradual redefinition of I_{der}, Π_p and Π_{sel} which relates to the concept of scaffolding.

The formalism is applied to a sequence learning task in which the introduced concepts are illustrated with focus on how bias of various kinds can be used to enable learning from a single demonstration. The experiment shows how even a simple demonstration contains almost unavoidable ambiguities that have to be handled one way or another. In context of the presented formalism, these ambiguities appear clearly as a transition problem from behavior $B \subset I_{hist}$ to controller $\pi \in \Pi$, or as a controller selection problem in Π_{sel}. This research problem is believed to be crucial for the development of learning robots and is addressed in our ongoing research.

The presented work is an attempt to structure and formalize general principles and assumptions in LFD. Our aim is not to present the single best way to talk about behaviors, generalization, goals, and other LFD related concepts. Rather, we want to point out the importance of defining these concepts clearly. It is our hope that the presented work will provide useful insights to the mechanisms involved in LFD and thus contribute to further development of this powerful and promising area of robot learning.

Acknowledgments

The authors would like to thank Lars-Erik Janlert for many valuable comments on this paper, and Steven LaValle for inspiring discussions about information spaces.

Table 1. *Symbol index*

Symbol	Description	Definition
X	State space	Sec. 2.1
Y	Observation space	Sec. 2.2
$\tilde{Y}$	Observation history space	Sec. 2.3
$\tilde{y} \in \tilde{Y}$	Observation history	Eq. 2
U	Action space	Sec. 2.2
$\tilde{U}$	Action history space	Sec. 2.3
$\tilde{u} \in \tilde{U}$	Action history state	Eq. 3
h	Sensor function	Sec. 2.2
f	Action function	Sec. 2.2
Π	Controller space	Sec. 3
$\pi \in \Pi$	Controller	Eq. 1, Eq. 7
I	Information space	Sec. 2.3
$e \in I$	Sensory-motor event	Sec. 2.3
I_k	History information space (up to stage k)	Eq. 5
$\eta \in I_{hist}$	History information state	Eq. 4
$\eta_0 \in I_{hist}$	Initial conditions	Sec. 2.3
I_{hist}	Information history space (unbound)	Eq. 6
$B \subset I_{hist}$	Behavior	Sec. 2.5
$b \subset B$	Demonstration	Sec. 3
$R \subset I_{hist}$	Realization space	Sec. 3
λ	Learning function	Eq. 10
Λ	Realization function	Eq. 11
I_{der}	Derived information space	Sec. 3.2
$G \subset X$	Goal (defined in X)	Sec. 3.3
$G_I \subset I_{hist}$	Goal (defined in I_{hist})	Sec. 3.3
Π_p	Set of primitive controllers	Sec. 3.4
$\pi_p \in \Pi_p$	Primitive controller	Eq. 16
π_{sel}	Controller selection function	Eq. 17
κ	Information mapping	Eq. 12

References

[1] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Imitation with ALICE: learning to imitate corresponding actions across dissimilar embodiments. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 32:482–496, 2002.

[2] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Action, state and effect metrics for robot imitation. In 15th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN 2006), pages 232–237, Hatfield, September 2006.

[3] R. Amit and M. Matariċ. Parametric primitives for motor representation and control. In Int. Conf. on Robotics and Automation (ICRA), Washington DC, May 2002.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. Robotics and Autonomous Systems, 57(5):469–483, May 2009.

[5] R. C. Arkin. Behaviour-Based Robotics. MIT Press, 1998.

[6] P. Bakker and Y. Kuniyoshi. Robot see, robot do: an overview of robot imitation. In Proceedings of the AISB Workshop on Learning in Robots and Animals, pages 3–11, Brighton, 1996.

[7] D. Baldwin, A. Andersson, J. Saffran, and M. Meyer. Segmenting dynamic human action via statistical structure. Cognition, 106(3): 1382–1407, March 2008.

[8] D. C. Bentivegna. Learning from Observation using Primitives. PhD thesis, College of Computing, Georgia Institute of Technology, 2004.

[9] D. C. Bentivegna, C. G. Atkeson, and G. Cheng. Learning similar tasks from observation and practice. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2677–2683, Beijing, China, October 2006.

[10] A. Billard, Y. Epars, G. Cheng, and S. Schaal. Discovering imitation strategies through categorization of multi-dimensional data. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 3, pages 2398–2403 vol.3, 2003.

[11] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng. Discovering optimal imitation strategies. Robotics and Autonomous Systems, 47(2-3):69–77, June 2004.

[12] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, Handbook of Robotics. Springer, 2008.

[13] E. A. Billing. Cognition Reversed - Robot Learning from Demonstration. PhD thesis, Umeå University, Department of Computing Science, Umeå, Sweden, December 2009.

[14] E. A. Billing. Cognitive perspectives on robot behavior. In J. Filipe, A. Fred, and B. Sharp, editors, Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART), Special

Session LAMAS, pages 373–382, Valencia, Spain, January 2010.

[15] E. A. Billing and T. Hellström. Behavior recognition for segmentation of demonstrated tasks. In IEEE SMC International Conference on Distributed Human-Machine Systems, pages 228 – 234, Athens, Greece, March 2008.

[16] E. A. Billing, T. Hellström, and L. E. Janlert. Model-free learning from demonstration. In J. Filipe, A. Fred, and B. Sharp, editors, Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART), pages 62–71, Valencia, Spain, January 2010.

[17] E. A. Billing, T. Hellström, and L. E. Janlert. Behavior recognition for learning from demonstration. In Proceedings of IEEE International Conference on Robotics and Automation, Anchorage, Alaska, May 2010.

[18] C. Breazeal and B. Scassellati. Challanges in building robots that imitate people. In K. Dautenhahn and C. L. Nehahiv, editors, Imitation in Animals and Artifacts. MIT Press, 2002.

[19] C. Breazeal and B. Scassellati. Infant-like social interactions between a robot and a human caretaker. Adaptive Behavior, 8(1): 49–74, 1998.

[20] C. Breazeal and B. Scassellati. Robots that imitate humans. Trends in Cognitive Sciences, 6(11):481–487, November 2002.

[21] R. A. Brooks. New approaches to robotics. Science, 253(13): 1227–1232, 1991.

[22] R. W. Byrne and A. E. Russon. Learning by imitation: a hierarchical approach. The Journal of Behavioral and Brain Sciences, 16(3), 1998.

[23] S. Calinon and A. Billard. Recognition and reproduction of gestures using a probabilistic framework combining PCA, ICA and HMM. In Proceedings of the 22nd international conference on Machine learning, pages 105–112, Bonn, Germany, 2005. ACM.

[24] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation, 37(2):286–298, 2007.

[25] P. Cohen, N. Adams, and H. B. Voting experts: An unsupervised algorithm for segmenting. Intelligent Data Analysis, 11(6):607–625, 2007.

[26] A. Cypher, editor. Watch What I Do: Programming by Demonstration. MIT Press, 1993.

[27] T. S. Dahl. Behavior-Based Learning. PhD thesis, Faculty of Engineering, University of Bristol, UK, 2002.

[28] N. Delson and H. West. Robot programming by human demonstration: The use of human inconsistency in improving 3D robot trajectories. In Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94. Advanced Robotic Systems and the Real World, IROS '94., volume 2, pages 1248–1255, Munich, Germany, September 1994.

[29] J. Demiris and G. Hayes. Do robots ape? In Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents, pages 28–31, 1997.

[30] Y. Demiris and A. Dearden. From motor babbling to hierarchical learning by imitation: a robot developmental pathway. In Proceedings of the 5th International Workshop on Epigenetic Robotics, pages 31—37, 2005.

[31] Y. Demiris and M. Johnson. Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. Connection Science, 15(4):231–243, 2003.

[32] Y. Demiris and B. Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. Robotics and Autonomous Systems, 54(5):361–369, May 2006.

[33] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification (2nd Edition). Wiley-Interscience, 2001.

[34] A. Fod, M. Matariċ, and O. C. Jenkins. Automated derivation of primitives for movement classification. Autonomous Robots, pages 39–54, 2002.

[35] C. Giovannangeli and P. Gaussier. Human-Robot interactions as a cognitive catalyst for the learning of behavioral attractors. In 16th IEEE International Symposium on Robot and Human interactive Communication (RO-MAN 2007), pages 1028–1033, August 2007.

[36] S. F. Giszter, F. A. Mussa-Ivaldi, and E. Bizzi. Convergent force field organized in the frog's spinal cord. Journal of Neuroscience, 13(2):467–491, 1993.

[37] J. G. Greeno. Special issue on situated action. In Cognitive Science, volume 17, pages 1–147. Ablex Publishing Corporation, Norwood, New Jersey, 1993.

[38] F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement learning for imitating constrained reaching movements. RSJ Advanced Robotics, Special Issue on Imitative Robots, 21(13): 1521–1544, 2007.

[39] M. Haruno, D. M. Wolpert, and M. M. Kawato. MOSAIC model for sensorimotor learning and control. Neural Comput., 13(10): 2201–2220, 2001.

[40] M. Haruno, D. M. Wolpert, and M. Kawato. Hierarchical MOSAIC for movement generation. In International Congress Series 1250, pages 575– 590. Elsevier Science B.V., 2003.

[41] T. Hastie, R. Tibshirani, and J. H Friedman. The Elements of Statistical Learning. Springer, August 2001.

[42] T. Hellström. Teaching a robot to behave like a cockroach. In Proceedings of the Third International Symposium on Imitation in Animals and Artifacts in Hatfield UK, pages 54–61, 2005.

[43] T. Hellström, T. Johansson, and O. Ringdahl. Development of an autonomous forest machine for path tracking. In P. Corke and S. Sukkariah, editors, Field and Service Robotics - Results of the 5th International Conference FSR, volume 25 of Springer Tracts in Advanced Robotics, pages 603–614. Springer, 2006.

[44] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. Proceedings of IEEE Transactions on Robotics, 24(6): 1463–1467, 2008.

[45] E. Hutchins. Cognition in the Wild. MIT Press, Cambridge, Massachusetts, 1995.

[46] R. A. Peters II and C. L. Campbell. Robonaut task learning through teleoperation. In Proceedings of the 2003 IEEE, International Conference on Robotics and Automation, pages 23–27, Taipei, Taiwan, September 2003.

[47] L. E. Janlert. Modeling change - the frame problem. In Z. W. Pylyshyn, editor, The Robot's Dilemma, pages 1 – 41. Ablex Publishing, Norwood, New Jersey, 1987.

[48] K-Team. Khepera robot. http://www.k-team.com, 2007.

[49] H. Kadone and Y. Nakamura. Segmentation, memorization, recognition and abstraction of humanoid motions based on correlations and associative memory. In Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots, pages 1–6, University of Genova, Genova, Italy, 2006.

[50] H. Kadone and Y. Nakamura. Symbolic memory for humanoid robots using hierarchical bifurcations of attractors in nonmonotonic neural networks. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2900–2905, Edmonton, AB, Canada, 2005.

[51] N. Koenig and M. J. Matariċ. Behavior-Based segmentation of demonstrated tasks. In International Conference on Development and Learning (ICDL), Bloomington, USA, May 2006.

[52] D Kuliċ, W Takano, and Y Nakamura. Incremental learning, clus-

tering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. The International Journal of Robotics Research, 27(7):761–784, July 2008.

[53] S. M. LaValle. Planning Algorithms. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[54] H. Lieberman, editor. Your Wish is My Command: Programming by Example. Morgan Kaufmann, San Francisco, 2001.

[55] L. Ljung. System Identification. Prentice-Hall, Simon & Schuster, Englewood Cliffs, New Jersey, 1987.

[56] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In National Conference on Artificial Intelligence (AAAI), pages 796–802, 1990.

[57] P. Martin and U. Nehmzow. Programming by teaching: Neural network control in the manchester mobile robot. In Proc. Intelligent Autonomous Vehicles, Helsinki. Springer Verlag, 1995.

[58] M. J. Matariċ. Behavior-Based control: Examples from navigation, learning, and group behavior. Journal of Experimental and Theoretical Artificial Intelligence, 9(2–3):323–336, 1997.

[59] M. J. Matariċ. Designing and understanding adaptive group behavior. Adaptive Behavior, 4(1):51–80, 1995.

[60] M. J. Matariċ. Integration of representation into Goal-Driven Behavior-Based robots. In IEEE Transactions on Robotics and Automation, volume 8, pages 304–312, 1992.

[61] M. J. Matariċ and M. J. Marjanovic. Synthesizing complex behaviors by composing simple primitives. In Proceedings of the European Conference on Artificial Life (ECAL-93), volume 2, pages 698–707, Brussels, Belgium, May 1993.

[62] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, Machine Intelligence 4, pages 463–502. Edinburgh University Press, 1969.

[63] T. M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers Computer Science Department Technical Report, New Brunswick, New Jersey, 1980.

[64] F. A. Mussa-Ivaldi and S. F. Giszter. Vector field approximation: a computational paradigm for motor control and learning. Biological cybernetics, 67:479–489, 1992.

[65] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi. Recognition and generation of leg primitive motions for dance imitation by a humanoid robot. In Proceedings of 2nd International Symposium on Adaptive Motion of Animals and Machines, Kyoto, Japan, 2003.

[66] C. L. Nehaniv and K. Dautenhahn. The correspondence problem. In K. Dautenhahn and C. L. Nehahiv, editors, Imitation in Animals and Artifacts. MIT Press, 2002.

[67] C. L. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris and A. Birk, editors, Learning Robots: An Interdisciplinary Approach, volume 24, pages 136–161. World Scientific Press, 2000.

[68] M. Nicolescu. A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains. PhD thesis, University of Southern California, 2003.

[69] A. Olenderski, M. Nicolescu, and S. Louis. Robot learning by demonstration using forward models of Schema-Based behaviors. In Proceedings of International Conference on Informatics in Control, Automation and Robotics, Barcelona, Spain, 2005.

[70] N. Otero, J. Saunders, K. Dautenhahn, and C. L. Nehaniv. Teaching robot companions: the role of scaffolding and event structuring. Connection Science, 20:111–134, June 2008.

[71] J. Peters and S. Schaal. Policy learning for motor skills. In Proceedings of 14th International Conference on Neural Information Processing (ICONIP 2007), pages 1–10, Berlin, Germany, November 2007. Springer.

[72] R. Pfeifer and C. Scheier. Sensory-motor coordination: the metaphor and beyond. Robotics and Autonomous Systems, 20 (2):157–178, June 1997.

[73] R. Pfeifer and C. Scheier. Understanding Intelligence. MIT Press. Cambrage, Massachusetts, 2001.

[74] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 578–585, 1993.

[75] B. Rohrer and S. Hulet. BECCA - a brain emulating cognition and control architecture. Technical report, Cybernetic Systems Integration Department, Univeristy of Sandria National Laboratories, Alberquerque, NM, USA, 2006.

[76] B. Rohrer and S. Hulet. A learning and control approach based on the human neuromotor system. In Proceedings of Biomedical Robotics and Biomechatronics, BioRob, 2006.

[77] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, NJ, 1995.

[78] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. Using Self-Imitation to direct learning. In 15th IEEE International Symposium on Robot and Human Interactive Communication, pages 244–250, 2006.

[79] J. Saunders, C. L. Nehaniv, K. Dautenhahn, and A. Alissandrakis. Self-Imitation and environmental scaffolding for robot teaching. International Journal of Advanced Robotics Systems, 4(1):109–124, 2007.

[80] B. Scassellati. Imitation and mechanisms of joint attention: A developmental structure for building social skills on a humanoid robot. Lecture Notes in Computer Science, 1562:176–195, 1999.

[81] H. A. Simon. The Sciences of the Artificial. MIT Press, Cambridge, Massachusetts, 1969.

[82] L. A. Suchman. Plans and Situated Actions. PhD thesis, Intelligent Systems Laboratory, Xerox Palo Alto Research Center, USA, 1987.

[83] J. Tani. On the interactions between top-down anticipation and bottom-up regression. Frontiers in Neurorobotics, 1:2, 2007.

[84] J. Tani and M. Ito. Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment. IEEE Trans. on Systems, Man, and Cybernetics Part A: Systems and Humans, 33(4):481–488, 2003.

[85] D. H. Wolpert and W. G Macready. No free lunch theorems for optimization. In IEEE Transactions on Evolutionary Computation, volume 1, pages 67–82, April 1997.

[86] D. M. Wolpert. A unifying computational framework for motor control and social interaction. Phil. Trans. R. Soc. Lond., B(358):593–602, March 2003.

[87] D. Wood, J. Bruner, and G. Ross. The role of tutoring in problem solving. Journal of Child Psychology and Psychiatry, 17:89–100, 1976.